DOCUMENT RESUME

ED 413 741 FL 024 762

AUTHOR	Erjavec, Tomaz
TITLE	Public Domain Generic Tools: An Overview.
PUB DATE	1995-00-00
NOTE	12p.; In: Language Resources for Language Technology: Proceedings of the TELRI (Trans-European Language Resources Infrastructure) European Seminar (1st, Tihany, Hungary, September 15-16, 1995); see FL 024 759.
PUB TYPE	Reports - Descriptive (141) -- Speeches/Meeting Papers (150)
EDRS PRICE	MF01/PC01 Plus Postage.
DESCRIPTORS	*Computational Linguistics; *Computer Software; *Discourse Analysis; Foreign Countries; Information Sources; Internet; Language Research; *Languages; Linguistic Theory; Standards; *Structural Analysis (Linguistics)
IDENTIFIERS	*Language Corpora; *Public Domain Software

ABSTRACT

This paper presents an introduction to language engineering software, especially for computerized language and text corpora. The focus of the paper is on small and relatively independent pieces of software designed for specific, often low-level language analysis tasks, and on tools in the public domain. Discussion begins with the application of standards to language corpora, and the role of information technology in promoting standardization. Current international standards and statistical tools are then examined briefly, and computational linguistic tools (morphological analyzers, implementation of formalisms, lexicon development environments) are noted. Problems in using public domain tools, and prospects for their resolution, are also discussed. A list of related World Wide Web resources is included. Contains 15 references. (MSE)

****
* Reproductions supplied by EDRS are the best that can be made *
* from the original document. *
****

ED 413 741

Public Domain Generic Tools: An Overview

Tomaž Erjavec

Language & Speech Group
Intelligent Systems Dept.
Jožef Stefan Institute
Ljubljana, Slovenia

PERMISSION TO REPRODUCE AND DISSEMINATE THIS MATERIAL HAS BEEN GRANTED BY

Natext
Vol3

TO THE EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)

U.S. DEPARTMENT OF EDUCATION
Office of Educational Research and Improvement
EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)

This document has been reproduced as received from the person or organization originating it.

Minor changes have been made to improve reproduction quality.

Points of view or opinions stated in this document do not necessarily represent official OERI position or policy.

f 004 760

BEST COPY AVAILABLE

1. Introduction

This paper gives an introduction to language engineering software, especially as it relates to computerised textual corpora. The focus of the paper is on language engineering tools, i.e. relatively small and independent pieces of software, meant for a particular, usually low-level task. Other, larger and more complex systems will be mentioned as well, as long as they are connected to the processing of textual material, in particular to corpus production and, to some extent, its utilisation. The paper does not discuss tools dealing with speech production or recognition although some of the corpus tools are relevant for producing speech corpora as well.

Even though the focus of this article will be on public domain tools, some software will be mentioned that does not, strictly speaking, belong in this category. There is a substantial variety of conditions that authors impose on their software, with proprietary, commercial products on one end, and freely available public domain software, that can be used for any purpose at the other end of the spectrum. Quite a few interesting linguistic tools fall somewhere between these two extremes with the most common conditions being that the software may be freely used for non-profit purposes only or that it falls under the GNU's general public license, which essentially forbids such software from being incorporated into proprietary programs. Furthermore, some authors request an explicit license to be signed before releasing their software. Nevertheless, such systems, even though not in the public domain, can be used by academic users and, in certain cases, can be of substantial use in an industrial environment as well. So, for example, if the software publicly released for academic use only is of sufficient interest, an arrangement can usually be made with the authors, or, if the software is a GNU library, it can still be used by proprietary software, as long as it does so in accordance with the GNU library general public license.

Using public domain tools has several obvious benefits, which are probably greatest for smaller research teams. These often lack funds to buy proprietary software or manpower for in-house development. Besides the obvious benefit of being for free, public domain tools allow for exploring a particular technology; even if a tool is not exactly what is required, the source code (where available) can be modified to suit particular needs. With public domain linguistic software that incorporates language particular resources, these resources (e.g., a morphological rule-base) can also be reused locally. Of course, the problems associated with using public domain tools should not be underestimated. These, along with some future prospects for their resolution will be discussed in the concluding section. Finally, for many tasks, in particular

for corpus work, commercial software is simply not available. Therefore, the available options are narrowed to using (and possibly modifying) public domain tools, or re-inventing the wheel by in-house development of the software.

The rest of the article is organised as follows: first, some introductory remarks are given on corpora and their connection with standards and technological advances; next the Unix platform, as the preferred development environment is discussed; this is followed by a section on SGML and statistical tools and a section on computational linguistic tools. Finally, some drawbacks to using public domain tools are given, together with recent efforts in this filed, concluding with a list or Web sites relevant to language engineering tools.

2. Corpora, Standards and the Internet

Recent years have seen a steep growth of computer corpora which have increased in size, number, and variety. Two of the more impressive examples of this trend are the hundred million annotated words of the British National Corpus and the hundred CD-ROMs of language resources offered by the Linguistic Data Consortium, which include annotated spoken corpora and multilingual corpora, both from a variety of sources.

The increased ability to produce and disseminate corpora is to a large extent due to technological advances. The dropping price and large capacities of mass storage media mean that large and heavily annotated corpora can be easily stored on-line. The growth in electronic communications, especially the success of the World Wide Web enables information on language resources or the resources themselves to be offered and accessed globally. In addition, the growing acceptance of certain standards that enable the exchange and platform independence of corpora encodings have also had a important impact on corpora availability and reuse. In particular, the Text Encoding Initiative guidelines (Sperberg-McQueen&Burnard, 1994, Ide&Veronis, 1995), which adopt the ISO standard SGML (Goldfarb, 1990) as their markup (meta)language are a significant contribution to the standardisation effort in this area.

This ease of availability and adoption of standards is important not only for corpora themselves, but also for software that helps in producing and, to a lesser extent, utilising these corpora. Internet connections mean that such tools as are offered to the public can be easily down-loaded or, in some cases, demonstrated, while the increasing adoption of standards minimises portability and interface problems.

3. The Unix platform

Although publicly available software does exist for PCs and Macs, it is Unix that is practically based on the notion of free software. This is due as much to the developmental history of Unix as to the GNU initiative of the Free Software Foundation (FSF). Although GNU software is not public domain, it can be used and modified freely, as long as it is not incorporated into proprietary systems. The utility of Unix as a development environment is due to it being a very powerful system, with a "amorphous" structure, that imposes relatively few constraints on program developers. For these reasons, it will be primarily Unix software that will be discussed in this article.

It should be noted, however, that it is also because of its power and reliance on free software that Unix is in many ways a troublesome system to use and maintain. Furthermore, Unix comes in a thousand and one flavours, depending on the exact platform in use (e.g., Solaris for SUN Sparcs, Linux for PCs, Irix for SGIs, etc.), thus, often making the installation of new programs a difficult undertaking.

I list next some software which runs on Unix (but often on other platforms as well) and can be of use in language engineering. First, Unix offers a variety of tools that do a specific job well, for example string or regular expression searching (**grep**) or sorting (**sort**). If programming languages can be thought of as "tools", then Unix offers a large selection of use in writing programs for language processing. The general purpose programming language which is currently, and probably for a while to come, the de facto standard is ANSI C and it's object-oriented extension **C++**. Unix also offers a number of languages that are particularly suited for string processing, such as **sed** (Doucherty, 1991) and **awk** (Aho et al., 1988). **Perl** deserves particular mention (Wall&Schwartz, 1991): it is suitable as much for writing short, throw-away programs as for complex conversion tasks. Finally, the GNU editor, Emacs, must be mentioned which, although as with most things with Unix, has a long learning curve, does offer very powerful functionality, and is freely extensible in its variant of Lisp.

4. SGML and Statistical Tools

The view of the corpus building process adopted here revolves around (presumably TEI conformant) SGML as the underlying data representation format. The evolution of a corpus is seen as composed of three stages. The corpus texts will usually be obtained in some sort of machine readable legacy

format (e.g., an ASCII representation, RTF from Word files, etc.) which are first up-translated into a corpus-wide encoding format, i.e., SGML. This bibliographically and, to an extent, structurally encoded corpus is then usually additionally (SGML) annotated in a number of ways, for example, for part-of-speech or multilingual alignment. Finally, a corpus is utilised by *searching* and *rendering* its material, for example, by showing keywords in context that match a given criterion or by showing aligned multilingual texts side by side.

The Perl language is well suited for up-translation to SGML as well as most (non-linguistic) conversions of SGML documents; there also exists an SGML-aware Perl library perlSGML, written by Earl Hood. Another general purpose programming language which is in the public domain and particularly suited to the manipulation of character strings is Icon (Griswold&Griswald, 1990). It was developed at the University of Arizona, and was extensively used in the British National Corpus project.

The basic SGML tool is the validating parser that checks for syntactic well-formedness of SGML documents and reports errors in case the document is not well-formed. A number of such validators exist, quite a few of which are in the public domain, e.g., James Clark's **sgmls** and **sp**. The second essential "tool" that is needed is an SGML-aware editor. Most of these are commercial software; however, Emacs does have a special mode (**psgml**), meant for editing SGML files.

There also exist freely available programs specifically designed for conversion of SGML documents (although not designed for handling corpus data), for example, the Copenhagen SGML tool CoST and MID's **MetaMorphosis**. These allow for transformations of SGML documents and also for rendering SGML annotated data. These tools and others can be found in the various Internet SGML repositories.

While there are quite a few tools available for corpus development, the choice of corpus querying tools is much more limited. While some of the above tools might prove useful in designing such a system, an integrated corpus query system must combine speed, a powerful querying language, and a display engine. Some such systems have been developed for DOS, but they usually lacked support for non-English languages and relied on idiosyncratic corpus encoding schemes. One Unix system that is offered for research purposes is the Corpus Query System cqp/Xkwic by Stuttgart's Institute für Maschinelle Sprachverarbeitung (IMS). The corpus query processor **cqp** is a command-language based query interpreter, which can be used independently or by **Xkwic**, which is a X-windows graphical user interface.

The last part of this section mentions some statistical linguistic tools used for corpus annotation. Part-of-speech taggers take as their input a word-form

together with all its possible morphosyntactic interpretations and output its most likely interpretation, given the context in which the word-form appears. So, for example, the word-form "tags" by itself can be interpreted as the plural noun or as the third person singular verb, whereas in the context "it tags word-forms" only the verb interpretation is correct. While syntactic parsers also perform such disambiguation, pure rule-based approaches tend to have low coverage and speed, and the investment into building rulesets for a particular language is prohibitive.

Recently there has been an increased interest in statistically based part-of-speech taggers, which use the local context of a word form for morphosyntactic disambiguation. Such taggers have the advantage of being fast and can be automatically trained on a pretagged corpus. Their success rate depends on many factors, but is usually at or below 96%. Two better known such taggers in the public domain are the Markov model-based Xerox tagger written in Lisp (Cutting et al., 1992) and Brill's rule-based tagger in C (Brill, 1992).

Finally, another pure statistical tool is the Gale and Church aligner, (Gale&Church, 1993) which sentence-aligns a text and its translation. It produces surprisingly good results by very simple means as it incorporates no linguistic knowledge but makes use of the basic insight that a text and its translation will have roughly the same number of characters.

5. Computational linguistic tools

This section deals with software that belongs to computational linguistics proper and includes morphological analysers, implementations of formalisms, and lexicon development environments. These systems can hardly be considered "tools" as they are often large and complex. They are, furthermore, only distantly connected to corpus development or exploitation. Nevertheless, they provide an environment for advanced language engineering tasks (e.g., machine translation) and it would be remiss not to mention them.

For morphological analysis and synthesis, Koskenniemi's finite-state *two-level model* is by far the most widely used and investigated. It is primarily meant to deal with spelling changes at or near morpheme boundaries. The best known implementation is probably PC-KIMMO (Antworth, 1990) although a number of other implementations also exist. Information about them, as well as about other systems, not based on the two-level model is available from the Saarbrücken's DFKI Natural Language Software Registry.

For general lexical structuring, including, but not limited to morphological dependencies, a simple, yet powerful and efficient language is DATR

(Evans&Gazdar, 1990). DATR is a lexical knowledge representation language in which it is possible to define networks allowing multiple default inheritance. The original Sussex version, which is publicly available, is implemented in Prolog.

Syntactic parsing, which usually forms the basis of more advanced language engineering applications has probably been the subject of most research in computational linguistics. It is, therefore, not surprising that a host of (public domain) parsing programs are available. For example, Prolog implementations usually offer a Definite Clause Grammar (**DCG**) module, and a number of various parsers (chart, Tomita, etc.) are available via the Internet, e.g., via **DFKI**.

Apart from **DCG**, the best known unification-based context-free parser is the **PATR** system (Shieber et al., 1983). More recent unification-based systems have replaced untyped feature structures of PATR with typed ones, thus, conferring the benefits of type checking and type inheritance to their grammars. Given that these systems can be used for other purposes apart from just parsing (e.g., machine translation), they are better classified as implementations of linguistic formalisms. There are a number of such systems available, pointers to which can be, again, found at the DFKI Web page. Here we will mention only three of the better known ones. The Attribute Logic Engine (ALE) (Carpenter&Penn, 1994) is written in Prolog and incorporates a chart parser and lexical rules. It is optimised for speed of processing which, however, makes it less than ideal for a grammar development environment. IMS offers two systems: Comprehensive Unification Grammar (CUF) (Dörre&Eisele, 1991) written in Prolog and Typed Feature Formalism (TFS) (Zajac, 1992) in Lisp. Especially CUF offers a very powerful grammar development environment; for a detailed comparison between ALE, CUF, and TFS, see also (Manandhar, 1993). Finally, it should be noted that of the three systems, ALE is available in source code, the other two being distributed in their compiled version only.

6. Drawbacks and Prospects

The penalties of using public domain tools should not be underestimated: the tools often do not come with all the bugs ironed out, with a detailed documentation or with the exact functionality required on the platform that we have. Maintenance is also often lacking as the developers' interests can have turned to other areas and support is, of course, a voluntary effort and cannot be counted on. For the field of multilingual language engineering, an

especially serious problem is that most of the available linguistic engineering software (public domain or commercial) to date was written for the English language or at best for (major) European Union (EU) languages. This bias gives rise to problems with "foreign" character sets, collating sequences, and the format of dates, numbers, and the like.

A connected problem is the lack of standards or, in some cases, conflicting standards for software development. This gives rise to tools that are often incompatible with one another, e.g., by virtue having different input/output formats and protocols. This can make their integration a daunting task, requiring extensive modifications of the tools. This is not to say that standards concerning software development have not been developed or are being considered by various institutions, e.g., by ISO and FSF. The work that is specifically addressed towards linguistic engineering are the *Guidelines for Linguistic Software Development*, which are being produced as a joint effort of the EU sponsored MULTEXT and MULTEXT-East projects and the Eagles sub-group on Tools, established in spring 1995. In particular, these guidelines are to address questions of usability, portability, compatibility and extensibility of linguistic software, concentrating in the first place on the Unix environment.

A number of other European Union projects have been concerned with developing linguistic software. However, in most cases the produced software is proprietary and, hence, not publicly available. A notable exception is the MULTEXT(-East) (this volume) project, which aims to make freely available to the academic community a number of SGML-based corpus processing tools. These include re-implementations of the already mentioned Xerox tagger, the Gale & Church aligner and a morphological synthesiser based on the two-level model.

For obtaining the tools mentioned in this paper, as well as a host of others, the easiest way is via the Web. A number of Web sites that provide further pointers to resources of interest to linguistic engineering, already exist: some of them are listed below. While some are the product of voluntary effort by individuals, there are also official bodies that disseminate information via the Internet, e.g., the DFKI Natural Language Software Registry or the EU Relator project. In connection with this, the pioneering effort of Edinburgh's Language Technology Group should also be mentioned: LTG offers a *Language Software Helpdesk*, which is a free service dedicated to the support of public domain, and freely available software for natural language processing and the fostering of its use in practical applications.

Finally, the TELRI Concerted Action also has a working group on "Lingware Dissemination". Its purpose is to increase the availability of language

engineering tools by making available, via the Web, information on extant tools, by providing the public tools of TELRI partners, tools, and by improving such tools by adapting them to various languages and platforms.

7. WWW References

The following World-Wide Web pages can provide more information on many of the topics and tools introduced above:

SGML Web Page by Robin Cover and SGML Open:
http://www.sil.org/sgml/sgml.html
http://www.sgmlopen.org/

TEI home page:
http://www-tei.uic.edu/orgs/tei/

British National Corpus and Linguistic Data Consortium:
http://info.ox.ac.uk/bnc/
http://www.cis.upenn.edu/ldc/

GNU software ftp site (including Emacs, Perl, grep, etc.) and online documentation for GNU software:
ftp://prep.ai.mit.edu/pub/gnu/
http://www.ns.utk.edu/gnu/

SGML repository at Institute for Informatics, Oslo (including psgml, sgmls and sp) and Steve Pepper's Whirlwind Guide to SGML Tools:
ftp://ftp.ifi.uio.no/pub/SGML/
http://www.falch.no/people/pepper/sgmltool/

Taggers by Xerox and Brill:
ftp://parcftp.xerox.com/pub/tagger/
ftp://blaze.cs.jhu.edu/pub/brill/Programs/

DFKI Natural Language Software Registry and the IMS list of language engineering links:
http://cl-www.dfki.uni-sb.de/cl/registry/draft.html
http://www.ims.uni-stuttgart.de/info/FTPServer.html

SIL's Software (including PC-PATR, PC-KIMMO):
http://www.sil.org/computing/sil_computing.html

DATR ftp site:
ftp://ftp.cogs.sussex.ac.uk/pub/nlp/DATR/

ALE home page:
http://macduff.andrew.cmu.edu/ale/

IDS's Tools and resources, including CQP/Xkwic, CUF and TFS:
http://www.ims.uni-stuttgart.de/Tools/ToolsAndResources.html

LTG's Language Software Helpdesk:
http://www.ltg.hcrc.ed.ac.uk/projects/helpdesk/

MULTEXT with Eagles Guidelines for Linguistic Software Development and MULTEXT-East:
http://www.lpl.univ-aix.fr/projects/multext/
http://www.lpl.univ-aix.fr/projects/multext-east/

Eagles and Relator:
http://www.ilc.pi.cnr.it/EAGLES/home.html
http://www.de.relator.research.ec.org:80/lg=en/index.mlhtml

TELRI and the Web version of this article:
http://www.ids-mannheim.de/telri/telri.html
http://nl.ijs.si/telri-wg5/pub-tools/

References

Aho, A.V., Kernighan, B.V., & Weinberger, P.J. 1988. "The Awk Programming Language". Addison Wesley, Reading, Massachussetts.

Antworth, E.L. 1990. "PC-KIMMO: a Two-level Processor for Morphological Analysis". No. 16 in Occasional Publications in Academic Computing. Summer Institute in Linguistics, Dallas, Texas.

Brill, E. 1992. "A Simple Rule-based Part of Speech Tagger". Proceedings of the Third Conference on Applied Natural Language Processing, ACL Trento, Italy.

Dörre, J. & Eisele, A. 1991. "A Comprehensive Unification-based Grammar Formalism". Technical Report Deliverable DYANA R3. 1.B, Centre for Cognitive Science, Edinburgh.

Doucherty, D. 1991. "sed & awk". O'Relly & Associates, Sebastopol, California.

Evans, R. and Gazdar, G. 1990. "The DATR Papers". Cognitive Science Research Paper CSRP-139, University of Sussex, Brighton.

Gale, W and Church, K.W. 1993. "Sentences in Bilingual Corpora". Computational Linguistics: 19(1), 75–102.

Goldfarb, C.F. 1990. The SGML Handbook Clarendon Press, Oxford.

Griswold, R.E. and M.T. Griswold, 1990. The Icon Programming Language (second edition). Prentice Hall, New Jersey.

Ide, N. and J. Véronis, (eds). 1995. The Text Encoding Initiative: Background and Context. Kluwer Academic Publishers, Dordrecht.

Manandhar, S. 1993. "CUF in Context". Computational Aspects of Constraint-Based Lingiuistics Description. ILIC/Department of Philosophy, University of Amsterdam.

Shieber, S., H. Uszkoreit, J. Robinson, and M. Tyson, 1983. "The formalism and implementation of PATR-II". Research on Interactive Acquisition and Use of Knowledge, 39–79. AI Center, SRI International, Menlo Park, Cal.

Sperberg-McQueen, C.M. and L. Burnard, (eds). 1994. Guidelines for Electronic text Encoding and Interchange. Chicago and Oxford.

Wall, L. and R.L. Schwartz, 1991. "Programming Perl". O'Reilly & Associates, Sebastopol, California.

Zajac, R. 1992. "Inheritance and Constraint-based Grammar Formalism". Computational Linguistics, 18(2).